Exceptional service in the national interest

Sandia National Laboratories

Performance Portability in SPARC – Sandia's Hypersonic CFD Code for Next-Generation Platforms

23 Aug 2017 – DOE COE Performance Portability Meeting

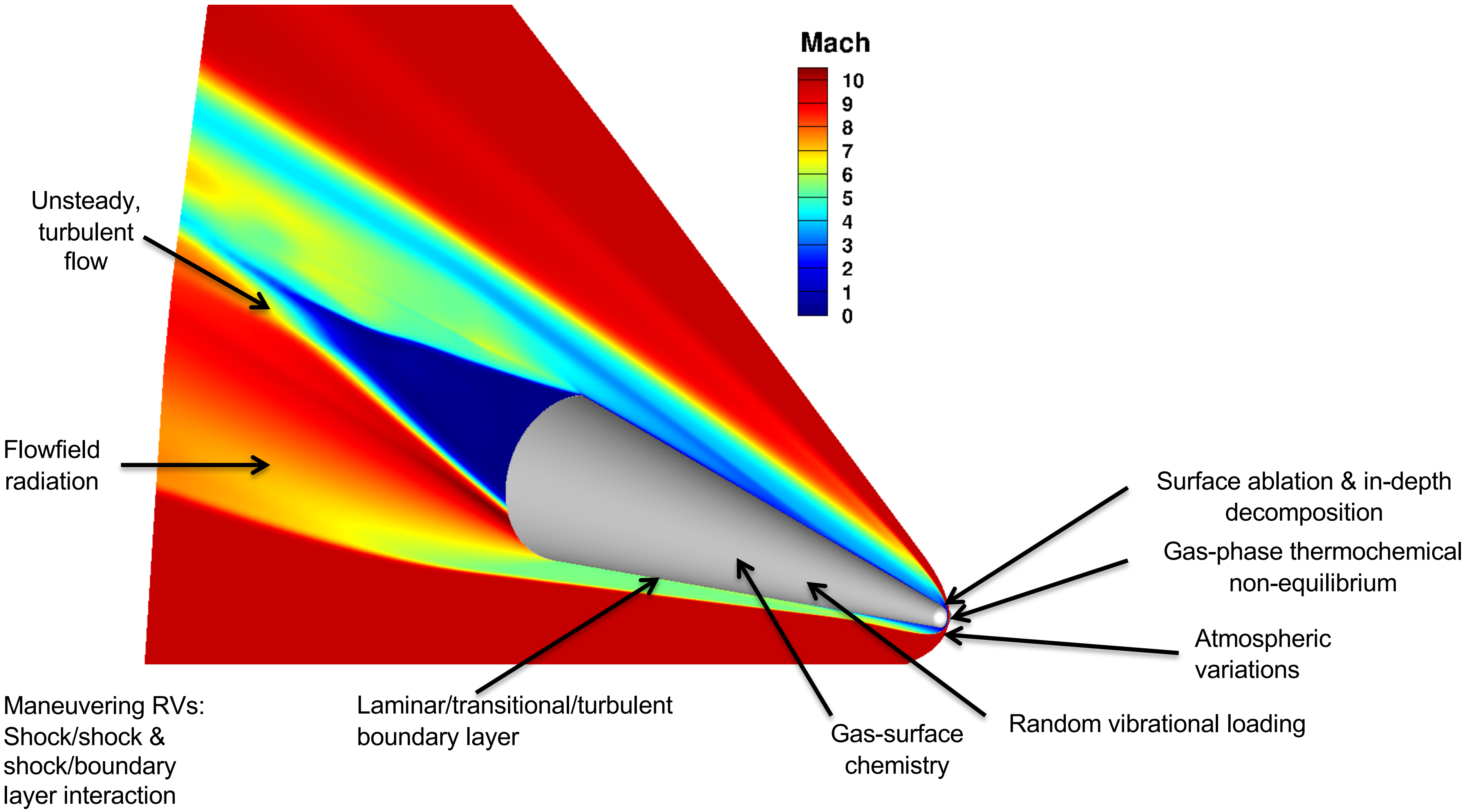
Micah Howard, SNL, Aerosciences Department
& the SPARC Development Team

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. SAND NO. 2017-8900 C

Motivation: Hypersonic Reentry Simulation



Unsteady, turbulent flow

Flowfield radiation

Surface ablation & in-depth decomposition

Gas-phase thermochemical non-equilibrium

Atmospheric variations

Maneuvering RVs: Shock/shock & shock/boundary layer interaction

Laminar/transitional/turbulent boundary layer

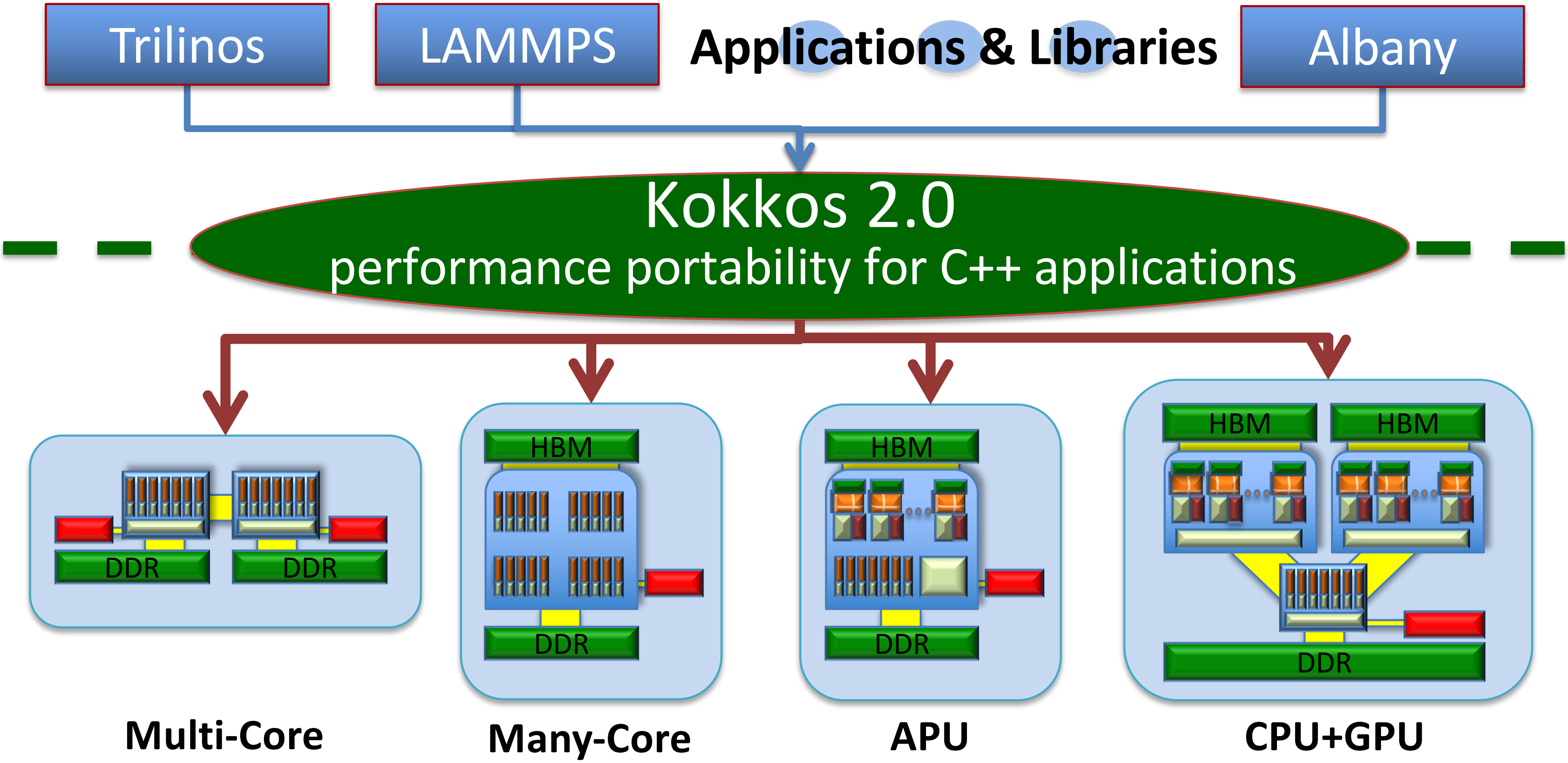
Gas-surface chemistry

Random vibrational loading

SPARC Compressible CFD Code

- State-of-the-art hypersonic CFD on next-gen platforms
  - Production: hybrid structured-unstructured finite volume methods
  - R&D: high order unstructured discontinuous collocation element methods
  - Perfect and thermo-chemical non-equilibrium gas models
  - RANS and hybrid RANS-LES turbulence models
- Enabling technologies
  - Scalable solvers
  - Embedded geometry & meshing
  - Embedded UQ and model calibration
- Credibility
  - Validation against wind tunnel and flight test data
  - Visibility and peer review by external hypersonics community
- Software quality
  - Rigorous regression, V&V and performance testing
  - Software design review and code review culture

Performance Portability - Kokkos


Trilinos

LAMMPS

Applications & Libraries

Albany

Kokkos 2.0

performance portability for C++ applications

Multi-Core

Many-Core

APU

CPU+GPU

Performance Portability

The problem on Heterogenous Architectures (e.g. ATS-2)

- C++ virtual functions (and function pointers) are not (easily) portable
- Answers?
    1. Kokkos support for portable virtual functions
    2. C++ standard support for portable virtual functions
    3. Run-time->compile-time polymorphism

SPARC has taken the ‘run-time->compile-time polymorphism’ approach

With this approach, we needed a mechanism to `dispatch` functions dynamically (run-time) or statically (compile-time)

Dynamic dispatch is possible on GPUs but requires the object be created for each thread or team on the GPU

Performance Portability

```
template <bool is_dyn, typename Type=MyClass>
struct Dispatcher {
 static void my_func (const MyClass* obj) {
  static_cast<const Type*>(obj)->Type::my_funcT();
};
```

Now we need a mechanism to convert run-time polymorphism to compile-time polymorphism so we can dispatch functions statically

Enter the `rt2ct` chain…

A “Create” chain is used to piece together compile-time instantiations of classes

The end of the chain (which is all compile-time) is handed to a Kokkos kernel

In this way, we can arbitrarily handle combinations of physics models (GasModels, FluxFunctions, BoundaryConditions) for (efficient) execution on GPUs

Threaded Assembly/Solves

Threaded Assembly on Structured Grids: `MeshTraverserKernel`

`MeshTraverserKernel` allows a physics code (think flux/flux Jacobian computation and assembly) to operate on a structured (`i,j,k`) block

- implements a multi-dimensional range policy for `Kokkos::parallel_for`
- provides `i,j,k` line traversal (CPU/KNL) and ‘tile’ traversal (GPU)

```
class PhysicsKernel :
  public MeshTraverserKernel<PhysicsKernel>
{ /* ... */ };
```

`Array4D` node-level multi-dimensional data for a structured block

- wraps a `Kokkos::DualView`

Graph coloring (red-black) to avoid atomics during assembly

Threaded solves provided through `Tpetra/Belos` (point-implicit, GMRES)

- OpenMP used for SPARC’s native point-implicit and line-implicit solvers

Net result of FY16 work: SPARC is running, end-to-end,
(equation assembly + solve) on the GPU

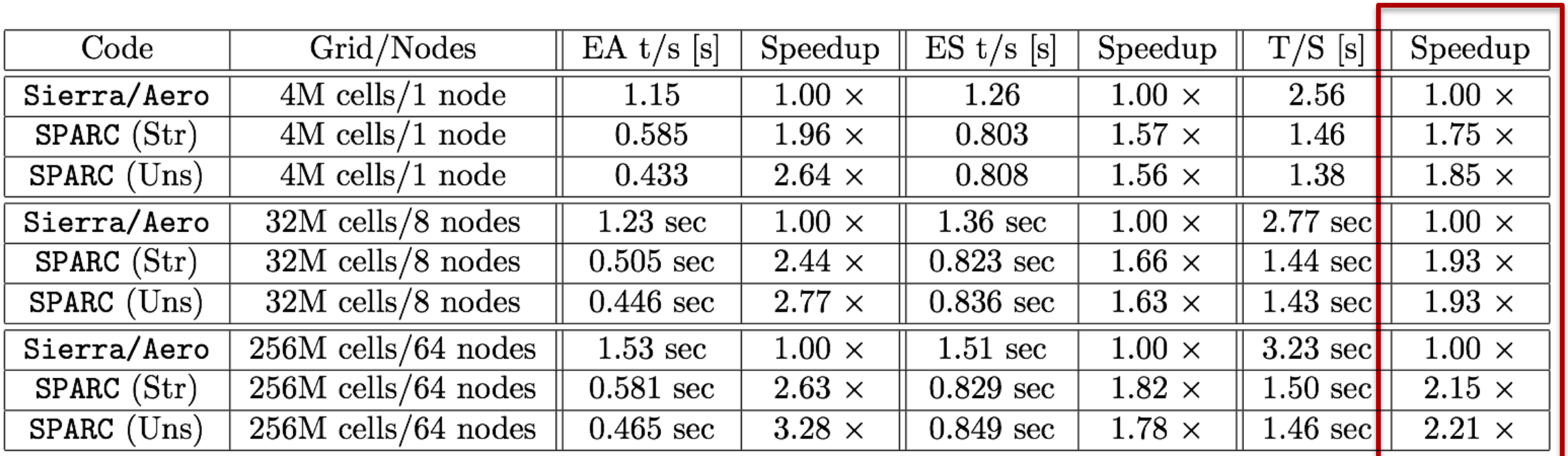
Performance Portability

- SPARC is running on all testbed, capacity & capability platforms available to SNL, notably:
  - Knights Landing (KNL) testbed
  - Power8+GPU testbed
  - Sandy Bridge & Broadwell CPU-based ‘commodity clusters’
  - ATS-1 – Trinity (both Haswell and KNL partitions)
  - ATS-2 – Power8+P100 ‘early access’ system

SPARC vs Sierra/Aero Performance

For the Generic Reentry Vehicle use-case…

Investigation of CPU-only, MPI-only performance



Code	Grid/Nodes	EA t/s [s]	Speedup	ES t/s [s]	Speedup	T/S [s]	Speedup
Sierra/Aero	4M cells/1 node	1.15	1.00 ×	1.26	1.00 ×	2.56	1.00 ×
SPARC (Str)	4M cells/1 node	0.585	1.96 ×	0.803	1.57 ×	1.46	1.75 ×
SPARC (Uns)	4M cells/1 node	0.433	2.64 ×	0.808	1.56 ×	1.38	1.85 ×
Sierra/Aero	32M cells/8 nodes	1.23 sec	1.00 ×	1.36 sec	1.00 ×	2.77 sec	1.00 ×
SPARC (Str)	32M cells/8 nodes	0.505 sec	2.44 ×	0.823 sec	1.66 ×	1.44 sec	1.93 ×
SPARC (Uns)	32M cells/8 nodes	0.446 sec	2.77 ×	0.836 sec	1.63 ×	1.43 sec	1.93 ×
Sierra/Aero	256M cells/64 nodes	1.53 sec	1.00 ×	1.51 sec	1.00 ×	3.23 sec	1.00 ×
SPARC (Str)	256M cells/64 nodes	0.581 sec	2.63 ×	0.829 sec	1.82 ×	1.50 sec	2.15 ×
SPARC (Uns)	256M cells/64 nodes	0.465 sec	3.28 ×	0.849 sec	1.78 ×	1.46 sec	2.21 ×

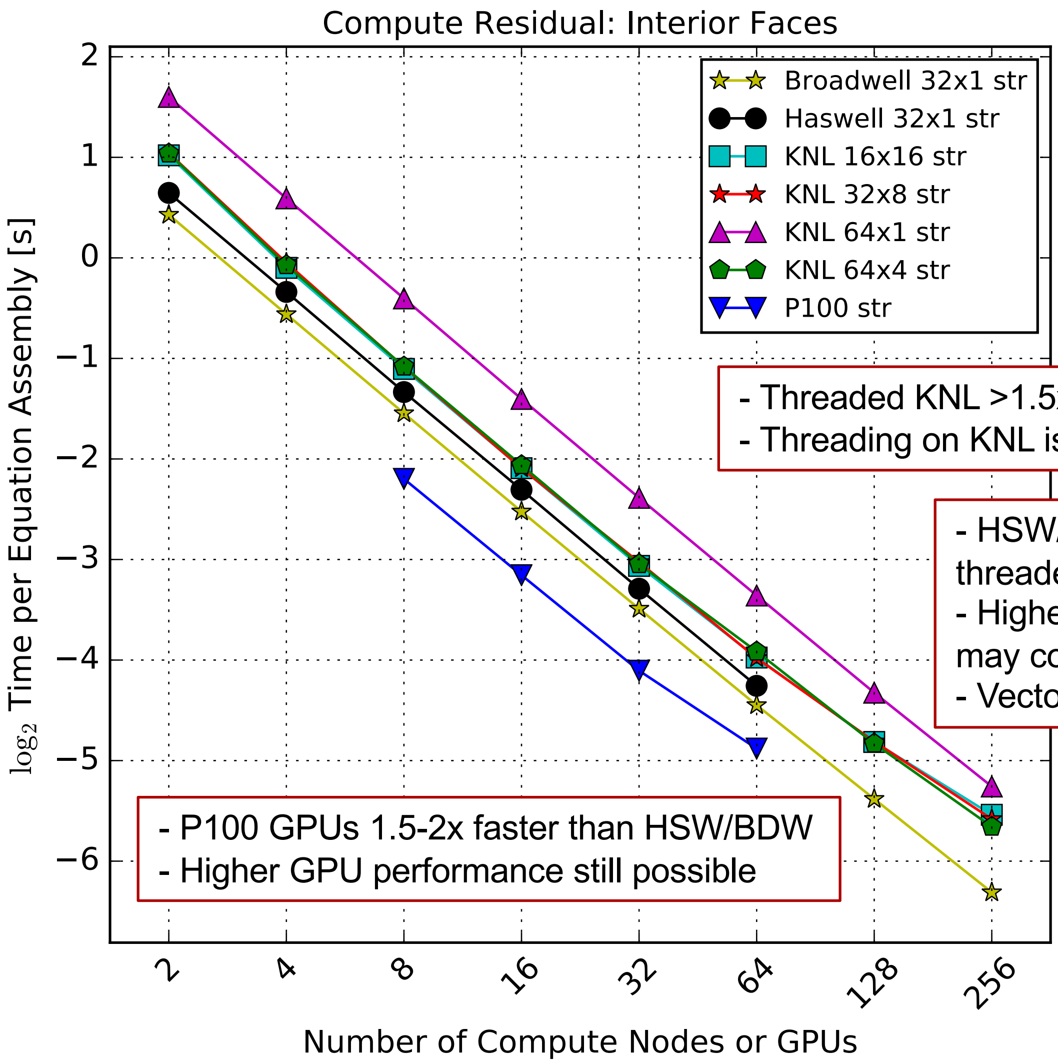
(EA t/s = Equation Assembly time/step; ES t/s = Equation Solve time/step; T/S = Total Time/Step)

- SPARC performing ~2x faster than Sierra/Aero
- Parallel efficiency is better than Sierra/Aero
- Even higher performance from SPARC for CPU-only systems will come with continued investment in NGP performance optimization
- Structured vs unstructured performance…

SPARC: Strong Scaling Analysis

For the heaviest kernel during equation assembly…

First…
lower =
faster
&
this is a
log2 scale



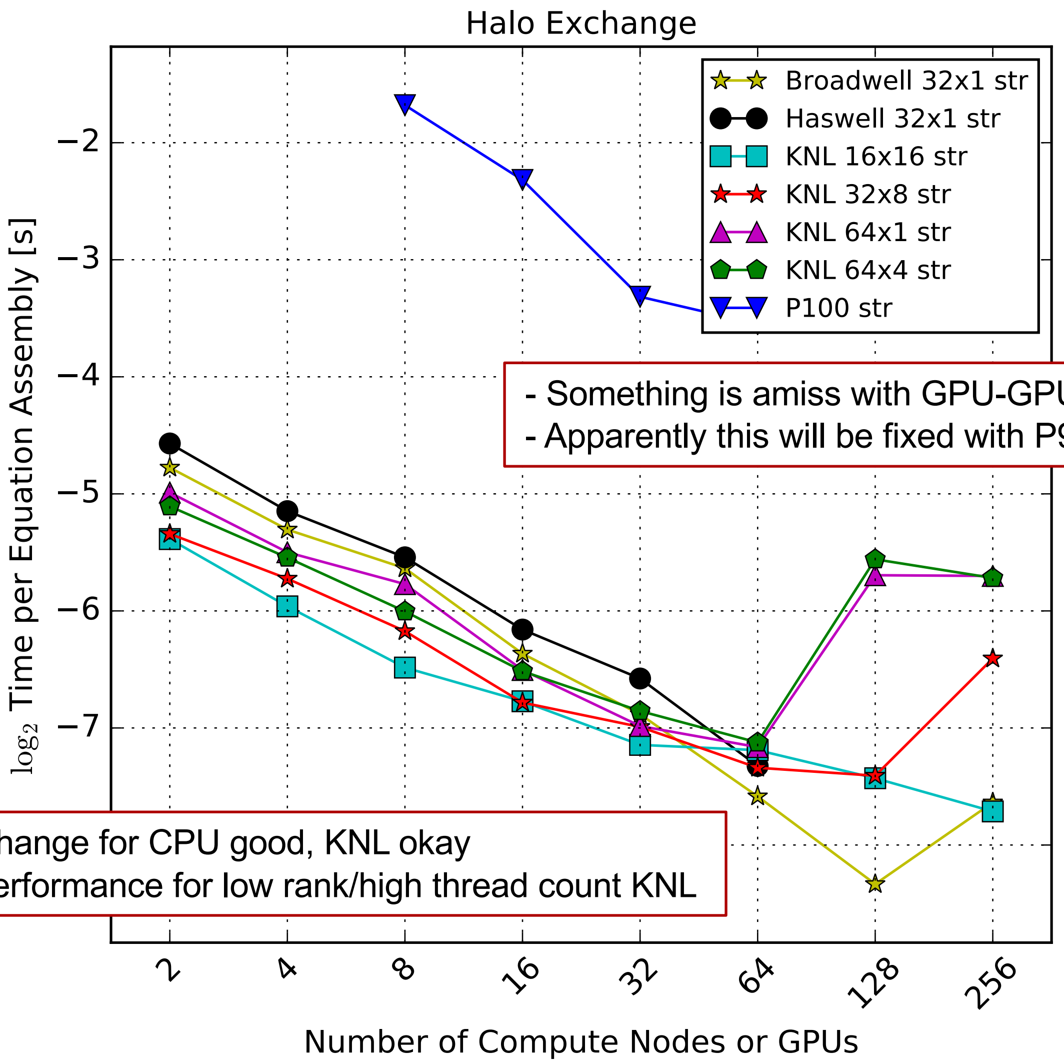
- Threaded KNL >1.5x faster than MPI-only KNL
- Threading on KNL is important

- HSW/BDW 1.25-1.5x faster than threaded KNL
- Higher KNL assembly performance may come from SIMD vectorization
- Vectorization a FY18 deliverable

- P100 GPUs 1.5-2x faster than HSW/BDW
- Higher GPU performance still possible

SPARC: Strong Scaling Analysis

For one critical MPI communication during equation assembly…

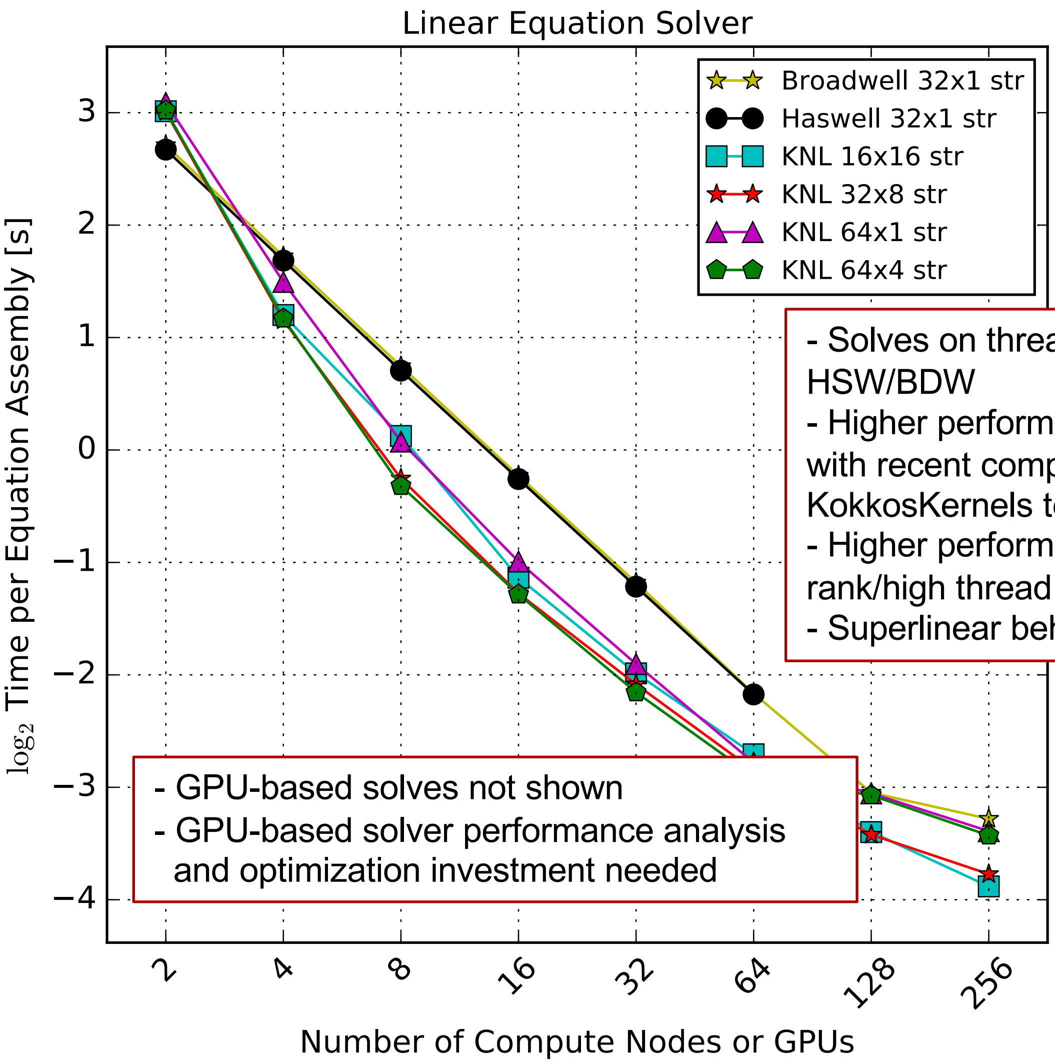


- Something is amiss with GPU-GPU MPI on P8/P100 systems
- Apparently this will be fixed with P9/Volta?

- Halo exchange for CPU good, KNL okay
- Higher performance for low rank/high thread count KNL

SPARC: Strong Scaling Analysis

For the linear equation solve…



- Solves on threaded KNL ~2x faster than HSW/BDW
- Higher performance on KNL still possible with recent compact BLAS work by the KokkosKernels team
- Higher performance at scale for low rank/high thread count KNL
- Superlinear behavior a DDR/HBM effect

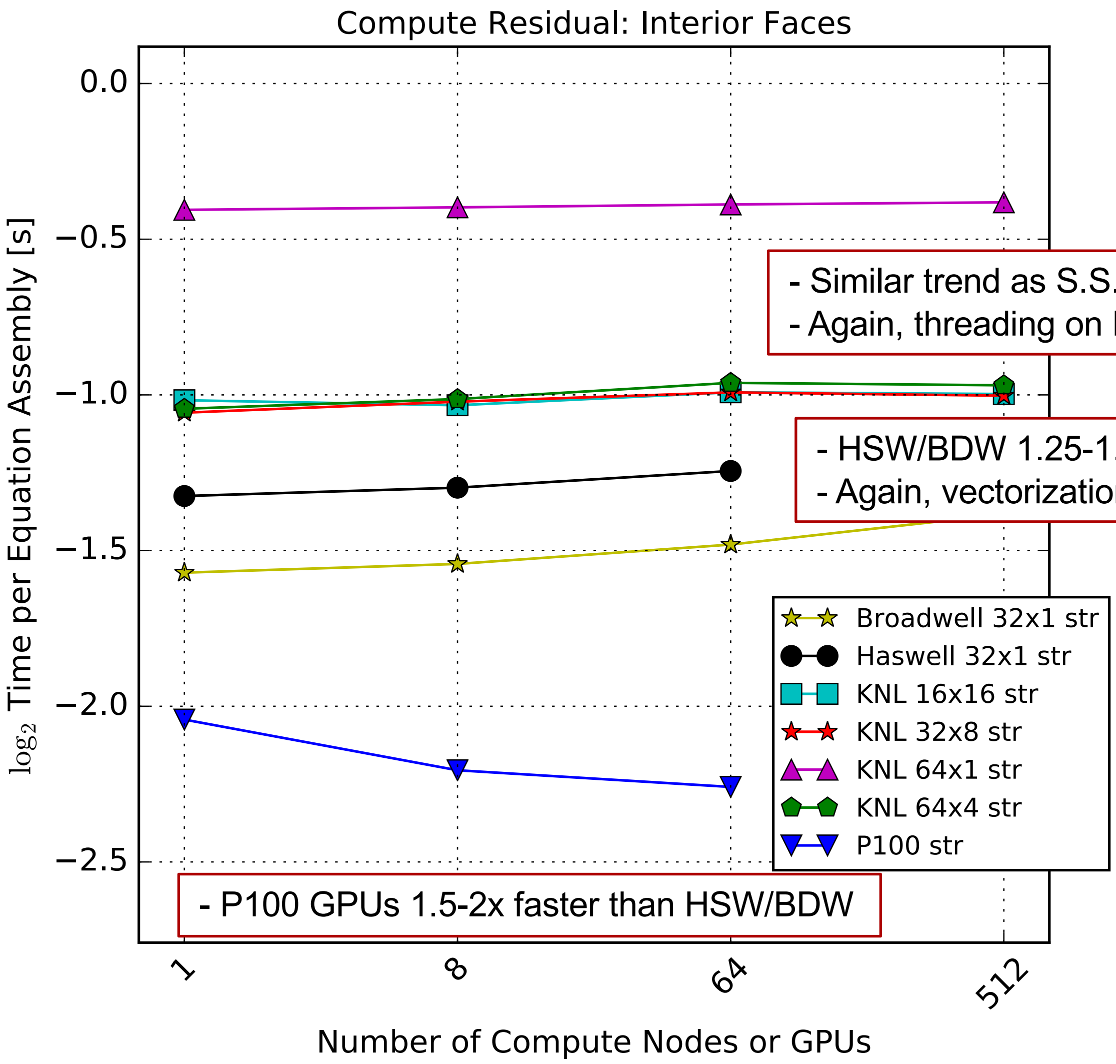
- GPU-based solves not shown
- GPU-based solver performance analysis and optimization investment needed

SPARC: Weak Scaling Analysis

For the heaviest kernel during equation assembly…

Recall…
lower =
faster
&
this is a
log2 scale

Compute Residual: Interior Faces



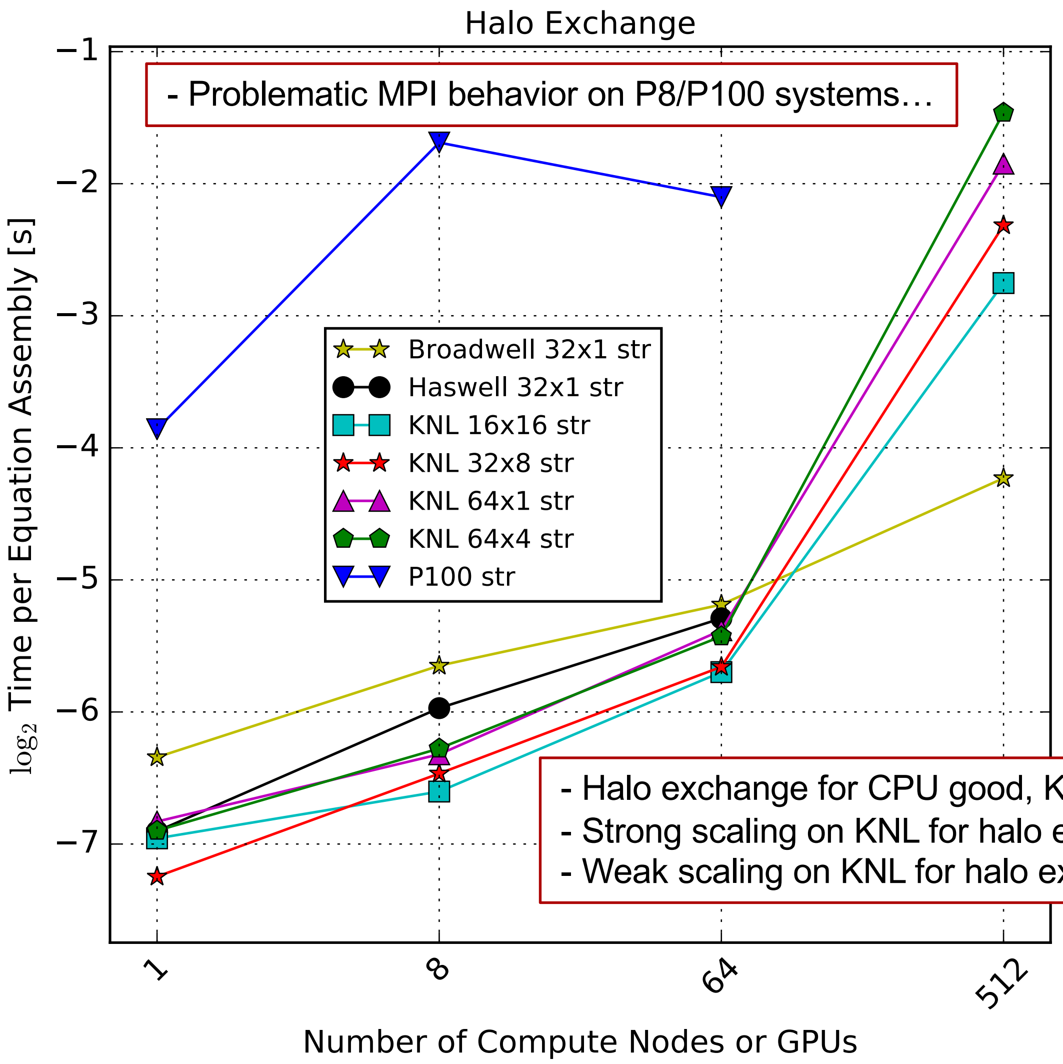
- Similar trend as S.S.: Threaded KNL >1.5x faster
- Again, threading on KNL is important

- HSW/BDW 1.25-1.5x faster than threaded KNL
- Again, vectorization may help

- P100 GPUs 1.5-2x faster than HSW/BDW

SPARC: Weak Scaling Analysis

For one critical MPI communication during equation assembly…

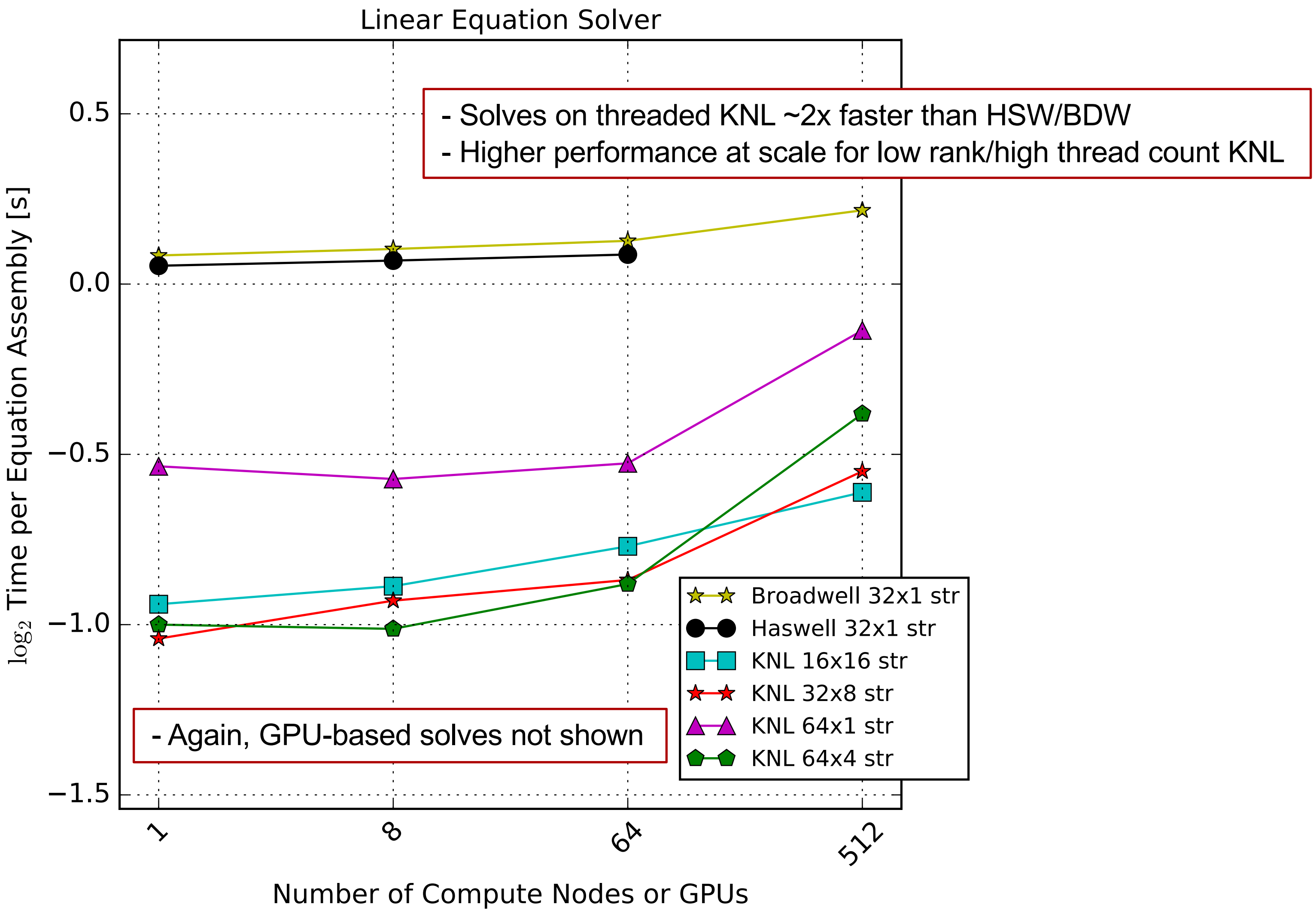


- Problematic MPI behavior on P8/P100 systems…

- Halo exchange for CPU good, KNL okay
- Strong scaling on KNL for halo exchange was okay
- Weak scaling on KNL for halo exchange needs investigation

SPARC: Weak Scaling Analysis

For the linear equation solve…



- Solves on threaded KNL ~2x faster than HSW/BDW
- Higher performance at scale for low rank/high thread count KNL

- Again, GPU-based solves not shown

Positive and Negative Experiences Developing PP Code

- Positive
  - Kokkos has helped – we can claim we are mostly portable and performant
    - The amount of architecture specific code is insignificant
  - Faster code has resulted from designing performance portable code
- Negative
  - The DevOps challenge: building the codes and its dependent libraries on several disparate architectures and establishing testing
  - The Performance analysis challenge: collecting meaningful performance data on each architecture
  - Developing for the GPU:
    - This drove a lot of our software design decisions
    - A CPU-only build on a Linux workstation with Intel compiler:
      - Clean: 13 min; Incremental: 3 min
    - A GPU build on a Power8 node with GCC/Cuda8 compiler:
      - Clean: 47 min, Incremental: 40 min

Summary

- SPARC is being developed as a performance portable compressible CFD code to address the challenges posed by next-generation computing platforms

- ‘The good’ for performance portability and SPARC:
  - CPU-only, MPI-only performance is ~2x faster than the reference code
  - Linear solves are ~2x faster for threaded KNL than CPU
  - Most significant assembly kernels are ~2x faster for P100 than CPU

- Future work for performance portability and SPARC:
  - Improve assembly performance for KNL -> vectorization
  - Hope for the best for halo exchange on P9/Volta (and reduce our MPI comm)
  - Work on solver performance for GPUs